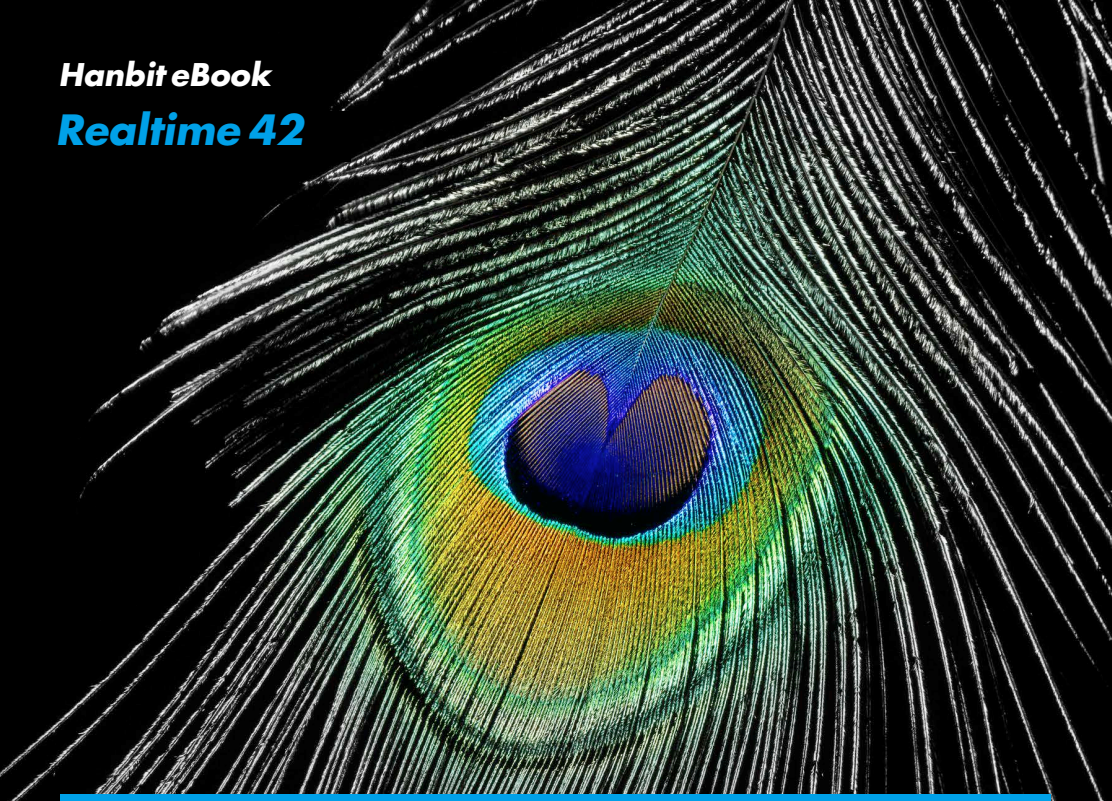


Hanbit eBook

Realtime 42

고객을 설득하는 데이터 시각화 실전 노하우

R로 하는 데이터 시각화

전희원 지음

한빛미디어 Hanbit Media, Inc.

R로 하는 데이터 시각화

고객을 설득하는 데이터 시각화 실전 노하우

지은이_ 전희원

7년 넘게 검색엔진·서비스 엔지니어로 재직하였으며 야후!를 거쳐 현재 SK텔레콤에서 데이터 분석 업무를 하고 있다. 직장생활 햇수와 비슷한 기간 동안 개인 블로그(http://freesearch.pe.kr)를 운영하고 있으며, 블로그를 통해서 다양한 생각을 공유하고 있다. 월간 마이크로소프트웨어, eWeek, IBM Developerworks에 검색, 데이터마이닝, 데이터 분석 관련 기고를 해왔으며 전자신문사와 ZDNet에서 R 관련 데이터 분석 실무와 데이터 마이닝 강좌를 진행하였다. 번역서로 『실전 예제로 살펴보는 집단지성 프로그래밍』(인사이트, 2010년)이 있다.

R로 하는 데이터 시각화 고객을 설득하는 데이터 시각화 실전 노하우

초판발행 2013년 10월 30일

지은이 전희원 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-649-4 15000 / **정가** 13,000원

책임편집 배용석 / **기획** 김병희 / **편집** 박세영
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 김현미
마케팅 박상용, 박주훈

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanb.co.kr / **이메일** ask@hanb.co.kr

Published by HANBIT Media, Inc. Printed in Korea

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanb.co.kr)로 보내주세요.

한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

저자 서문

지금으로부터 5년 전인 지난 2008년, 필자가 다니던 회사의 미국 본사 엔지니어가 위키(WIKI)에 올린 정체불명의 코드와 이미지를 보고 R이라는 언어를 처음 알게 됐다. 굉장히 생소한 문법을 사용하고, 간단한 코드로 시각화하고, 여러 통계량을 뽑아내는 모습을 보고 깊은 인상을 받았다.

이후 개발을 시작하기 전 파이썬으로 데이터에서 패턴을 뽑는 업무를 하면서 R을 틈틈이 사용하기 시작했고, 현재는 거의 풀타임으로 R과 함께 지내고 있다. 예전엔 파이썬 덕분에 일찍 퇴근할 수 있었다고 우스갯소리를 하곤 했는데, 요즘엔 R 덕분에 가설 검증을 빨리 할 수 있게 됐다고 이야기하곤 한다(물론, R 덕분에 일찍 퇴근할 수 있게 됐다).

데이터를 분석하는 아주 쉬운 툴부터 R처럼 문법을 가진 프로그래밍 언어까지 데이터를 다루는 환경은 다양하다. 그중에서도 필자가 R을 적극적으로 추천하는 이유는 데이터를 특정 측면으로 보기 위한 가장 빠른 길을 제시해 주고, 이에 따라 데이터를 자유롭게 분석할 수 있기 때문이다.

이런 자유로움은 가설 검증을 신속하게 하여 올바른 분석 방법을 빠른 시간에 찾도록 해준다. 물론 이런 이진검색binary search 적인 분석 접근 방법의 기초가 되는 건 역시 통계다.

R을 안 지 5년, 실제 실무에서 활용한 지 3년 정도가 됐다. 3년 동안 일과 시간은 물론 취미로도 R을 사용했으니 1만 시간 정도는 사용하지 않았나 싶다. 이젠 어느 정도 R의 동작 방식과 철학을 이해하게 되면서, 누군가와 R에 대한 정보를 나누고

싶다는 생각이 들어 이 책을 집필하기에 이르렀다. 그러면서 필자는 R을 처음 접하는 사람들이 지루한 통계 용어가 난무하는 내용이 아닌, 시각화라는 중요하면서도 재미있는 주제로 R과 첫만남을 했으면 좋겠다는 생각으로 이 책을 썼다.

아무쪼록 이 책이 어렵지 않고 재미있는 R 책으로 독자들에게 기억되길 바란다.

집필을 마무리 하며

전희원

대상 독자 및 예제 파일

초급 | 초중급 | **중급** | 중고급 | 고급

이 도서는 R을 처음 접하거나 시각화에 활용할 목적으로 R을 배우는 분들에게 적합하다. 이외에 R 시각화를 어느 정도 활용할 수 있는지 그 가능성을 가늠해 보려는 분에게도 유용할 것이다.

이 도서는 R을 사용해 시각화를 하는 데 필요한 제반 사항을 소개한다. 시각화가 왜 필요한지에 대한 소개와 더불어 시각화를 위한 데이터 전처리를 설명하고, 시각화의 핵심 부분은 ggplot2로 접근한다. 이후 그래프 후처리에 대한 내용은 Inkscape로 설명하고, 마지막으로 아름답고 섬세한 그래프를 만들기 위한 몇 가지 팁을 소개한다.

도서에서 사용한 예제 파일은 다음 웹 사이트에서 다운받을 수 있다.

- https://github.com/haven-jeon/R_based_visualization

도서에 있는 예제를 위한 소프트웨어

특정 결과를 공유할 때 'sessionInfo()'의 출력 결과를 사용한다. 이는 타인과 자신을 위해 매우 좋은 습관 중 하나다. 이 정보를 기반으로 해당 분석이 완벽하고 동일하게 돌아갈 수 있는지를 알 수 있기 때문이다. 따라서 'sessionInfo()' 결과를 공유한다. 아래 sessionInfo( ) 결과에 보이는 버전보다 높은 버전의 패키지와 R을 쓰는 한, 무리 없이 책의 예제 코드들이 동작할 것이다.

```
#책 예제를 실행한 환경정보
sessionInfo()
## R version 3.0.2 (2013-09-25)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
##
## locale:
## [1] LC_COLLATE = Korean_Korea.949 LC_CTYPE = Korean_Korea.949 LC_MONETARY = Korean_Korea.949
## [4] LC_NUMERIC = C LC_TIME = C
##
## attached base packages:
##  [1] stats4   grid   stats   graphics   grDevices   utils   datasets   methods   base
##
## other attached packages:
##  [1] shiny_0.7.0      ggmap_2.4     ggthemes_1.3.3     extrafont_0.15      devtools_1.3
##  [6] reshape2_1.2.2   party_1.0-9   modeltools_0.2-21  strucchange_1.4-7   sandwich_2.2-10
## [11] zoo_1.7-10       scales_0.2.3  gridExtra_0.9.1    data.table_1.8.10   sqldf_0.4-6.4
## [16] RSQLite.extfuns_0.0.1    RSQLite_0.11.4    chron_2.3-44    gsubfn_0.6-5    proto_0.3-10
## [21] DBI_0.2-7        plyr_1.8      ggplot2_0.9.3.1    xtable_1.7-1        knitr_1.2
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-6      caTools_1.14      coin_1.0-23       colorspace_1.2-3    dichromat_2.0-0
```

```
##  [6] digest_0.6.3      evaluate_0.5      extrafontdb_1.0  formatR_0.9          geosphere_1.2-28
## [11] gtable_0.1.2      httpuv_1.1.0      httr_0.2         labeling_0.2         lattice_0.20-23
## [16] mapproj_1.2-1     maps_2.3-6        MASS_7.3-29      memoise_0.1          munsell_0.4.2
## [21] mvtnorm_0.9-9996 parallel_3.0.2   png_0.1-6        RColorBrewer_1.0-5   RCurl_1.95-4.1
## [26] RgoogleMaps_1.2.0.5  rjson_0.2.13 RJSONIO_1.0-3    Rttf2pt1_1.2         sp_1.0-13
## [31] splines_3.0.2     tringr_0.6.2      survival_2.37-4  tcltk_3.0.2          tools_3.0.2
## [36] whisker_0.3-2
```

'#'로 시작하는 라인은 주석을, '##'로 시작하는 라인은 명령어 수행 결과를 각각 의미한다. 그 이외의 라인은 모두 소스코드다. 셸 프롬프트를 넣지 않은 이유는 책의 코드를 '복사/붙여넣기'해 손쉽게 코드를 수행할 수 있도록 하기 위해서다.

책을 읽기 전부터 위에 소개한 패키지 전부를 설치하지 않아도 된다. 책을 읽으면서 코드와 패키지를 소개할 때마다 차근차근 설치하면 된다. 대부분의 경우 문제가 생기지 않겠지만, 혹시 문제가 있다면 필자의 패키지 버전과 맞춰주면 해결될 것이다.

감사의 말

먼저 강력한 데이터 분석 언어인 R을 지금까지 만들고 가꿔온 R Core Team과 여러 패키지를 통해 데이터 분석의 혜안을 제시해준 ggplot2, plyr, reshape2 패키지 개발자인 해들리 위컴Hadley Wickham 교수님께도 존경과 감사의 마음을 드린다. 그리고 이 책에 소개한 모든 패키지의 개발자들께도 감사한다.

책을 쓸 엄두를 내지 못할 때, 작은 연재부터 시작하자고 용기를 주신 박세영 팀장님, 그리고 공개된 책을 전자책으로 내길 제안해주신 한빛미디어 김창수 팀장님과 김병희 대리님께 감사의 마음을 전하고 싶다.

집필 막바지에 자신도 R을 배우겠다며 책의 드래프트를 보면서 피드백을 준 아내 보현 씨에게 사랑과 감사의 마음을 전하며, 사랑스러운 두 아들 수빈·우빈, 그리고 부모님께 이 책을 바친다.

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 보다 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여 DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

1 | R 데이터 시각화의 시작

1.1 시각화의 중요성

최근 빅데이터 붐이 일어나면서 데이터 처리 플랫폼인 하둡Hadoop[01]과 함께 R[02]이라는 언어가 뜨고 있다. 그리고 빅데이터의 분석 방법으로 데이터 시각화data visualization가 다시 각광받고 있다. 빅데이터에서 데이터 시각화가 왜 각광받고 있을까?

빅데이터는 말 그대로 데이터가 매우 커서 개개의 데이터를 일일이 살펴보는 건 불가능에 가깝다. 평균, 표준편차 등 요약된 통계량이 분석가에게는 어느 정도 편리한 수단이라고 해도 이를 기반으로 누군가를 설득하기에는 충분하지 않다. 대부분의 사람이 공통적으로 갖고 있는 훌륭한 인지기관인 눈에 호소할 수 있는 데이터 시각화는 분석과 사람들을 설득하기 위한 효과적인 방법 중에 하나다. 아무리 데이터를 빨리 처리하는 하둡 플랫폼이 있다고 하더라도 결과적으로 분석 결과를 정보로 만들어 상대방에게 효과적으로 전달하지 않으면, 그저 거대한 데이터일 뿐이다.

필자는 모 금융 회사의 금융 데이터를 기반으로 빅데이터 플랫폼 프로젝트를 진행해본 경험이 있다. 이 경험을 토대로 이야기하자면, 빅데이터 처리 플랫폼이 아무리 빨리 데이터를 처리하더라도 결과 정보가 의미 있어야만 사람들이 빅데이터 플랫폼의 가치를 인정하였다. 그리고 정보의 의미를 보여주는 방법의 정점에 서 있는 것이 바로 '시각화 기술'이다.

프로젝트 진행 중 '많은 데이터를 처리한 결과, 우리가 얻는 정보나 혜택이 뭐냐?'는 질문을 받았다. 필자는 뒤에서 설명할 ggplot2[03]를 활용해 적절히 답변할 수 있

01 http://hadoop.apache.org/
02 참고 문헌 [14]를 참조(R Core Team, 2012)
03 참고 문헌 [17]을 참조(Wickham, 2009)

었다. 즉 시각화는 이러한 질문에 대해 답변을 바로 해줄 수 있는 몇 안 되는 기술 중 하나로, 빅데이터 플랫폼 엔지니어뿐만 아니라 데이터를 다루는 사람이 배워볼 가치가 충분히 있는 기술이다.

이제 본격적으로 시각화에 대해 알아보자.

1.2 시각화의 몇 가지 예

시각화의 한 예로 각 나라의 자살률을 살펴보자. 데이터는 위키피디아[04]에서 가져왔다(위키피디아의 데이터는 신빙성에 대한 논란이 있으니 참고하기 바란다).

> **NOTE_** 이 예제의 제시 이유는 시각화 학습을 위한 목적도 있으나, 한국의 자살률이 세계적으로 높다는 측면을 강조해 경각심을 키우기 위함도 있다. '다른 나라에 비해 우리나라의 자살률이 왜 높을까?'라는 의문을 독자가 갖는 순간, 아마도 이와 상관된 데이터를 찾아볼 수 있을 것이고 관련 있는 연구의 촉매가 될 수도 있을 거란 소망을 가져본다. 물론 그런 연구에서 시각화가 좋은 도구가 될 것임은 자명하다.

이 데이터에 대한 시각화에 부담이 있거나 시도해볼 여력이 없는 분들은 마이크로소프트 엑셀에서 10만 명당 몇 명이 자살을 하는지에 대해 내림차순으로 정렬해 어느 나라가 자살률이 높은지를 살펴보면 된다.

04 http://en.wikipedia.org/wiki/List_of_countries_by_suicide_rate

그림 1-1 엑셀로 본 자살률

	A	B	C	D	E	F
1	Rank	Country	Male	Female	Total	Year
2	1	Lithuania	61.3	10.4	34.1	2009
3	2	South Ko	41.4	21	31.2	2010
4	3	Guyana	39	13.4	26.4	2006
5	4	Kazakhsta	43	9.4	25.6	2008
6	5	Belarus			25.3	2010
7	6	Hungary	40	10.6	24.6	2009
8	7	Japan	33.5	14.6	23.8	2011
9	8	Latvia	40	8.2	22.9	2009
10	9	People's Republic of China			22.23	2010
11	10	Slovenia	34.6	9.4	21.9	2009

물론 다음과 같이(R에서 보여주는 자살률 데이터) 단순한 통계량으로만 보면 여자보다 남자의 자살률이 더 높다는 것을 알 수 있으나, 어떤 나라에서 그런 차이를 보이는지는 알 수 없다.

```
## Rank            Country                Male          Female          Total           Year
## Min.   : 1.0    Albania    : 1   Min    :0.00   Min.   : 0.00   Min.   : 0.00   Min.   :1978
## 1st Qu.: 27.5   Antigua and Barbuda: 1   1st Qu. : 5.45  1st Qu. : 1.70   1st Qu. : 3.60  1st Qu. :2006
## Median : 54.0   Argentina  : 1   Median :12.30   Median : 3.45   Median : 7.90   Median :2008
## Mean   : 54.0   Armenia    : 1   Mean   :14.39   Mean   : 4.14   Mean   : 9.62   Mean   :2006
## 3rd Qu.: 80.5   Australia  : 1   3rd Qu.:20.18   3rd Qu.: 6.05   3rd Qu.:14.20   3rd Qu.:2009
## Max.   :107.0   Austria    : 1   Max.   :61.30   Max.   :21.00   Max.   :34.10   Max.   :2011
##                 (Other)    : 101 NA's   :5       NA's   :5
```

위의 결과에서 얻을 수 있는 건 남녀의 자살률 차이뿐인데, 이를 그림 1-2처럼 '상자그림boxplot'[05]으로 표현하면 보기가 더 쉽다.

05 http://en.wikipedia.org/wiki/Box_plot

그림 1-2 자살률 데이터에 대한 상자그림 결과

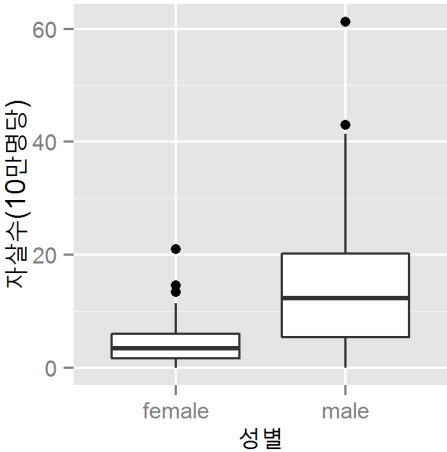
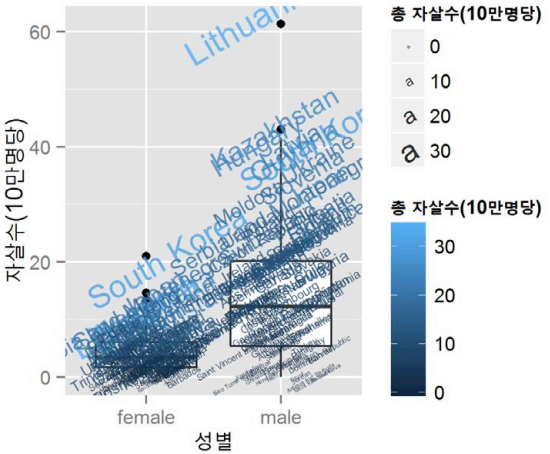


그림 1-2처럼 시각적으로 데이터를 표현함으로써 많은 정보를 한눈에 볼 수 있도록 한 것을 '데이터 시각화'라고 한다. 하지만 그림 1-2에도 뭔가 아쉬운 점이 있다. '아웃라이어outlier'[06]라는 평균이나 중앙값에서 멀리 떨어진 값이 어떤 것인지 표시해 주면, 훨씬 정보성이 올라갈 것이다.

그림 1-3와 같이 도식화하면 아웃라이어에 어떤 나라가 있는지 확인할 수 있다. 이 그래프만으로도 데이터 파일의 거의 모든 정보를 보여줄 수 있다. 텍스트 크기와 색깔로 10만 명당 자살자 수(남+여)의 상대적인 크기를 알 수 있으며, 남녀 개별적인 자살자 수도 볼 수 있어 상대적인 비교도 가능하다.

06 http://en.wikipedia.org/wiki/Outlier

그림 1-3 좀 더 자세한 자살률 데이터에 대한 상자그림 결과



다른 예로 필자가 블로그에 올려둔 '타이타닉 데이터 분석 포스팅'[07]을 살펴보자. 이 포스팅의 목적은 타이타닉호 사망자 통계를 기반으로 남자가 여자에 비해 사망할 확률이 높았음을 데이터를 통해 보여줘, 영화가 비극적인 결말로 끝날 가능성이 애초부터 높았음을 알려주는 것이었다. 이번 예제에서는 성별에 따른 생존율이 어떻게 다르고, 승객 등급에 따라 생존율 변화가 어떻게 되는지 가장 먼저 확인하고 싶어질 것이다. 일단 개별적으로 생존율을 확인하기 위한 다음과 같은 결과를 확인할 수 있을 것이다.

```
titanic.dt[, list(prob_as_class = length(which(survived == 1))/nrow(.SD)), by = pclass]
## pclass    prob_as_class
## 1: 1st           0.6192
## 2: 2nd           0.4296
## 3: 3rd           0.2553
```

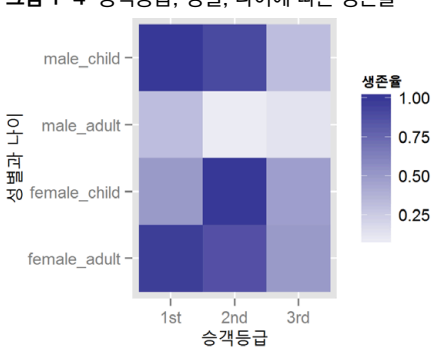
07 http://freesearch.pe.kr/archives/2855

```
titanic.dt[, list(prob_as_sex = length(which(survived == 1))/nrow(.SD)), by = sex]
##  sex      prob_as_sex
## 1: female      0.7275
## 2: male        0.1910

titanic.dt[, list(prob_as_sex = length(which(survived == 1))/nrow(.SD)), by = isminor]
##  isminor  prob_as_sex
## 1: adult       0.3658
## 2: child       0.5596
```

각 등급에 따른 생존율은 위와 같이 숫자로 표현할 수 있다(상세한 코드 설명은 시각화를 간단히 맛보려는 이번 장의 범위를 넘어가니 설명하지 않겠다). 이를 보면 승객등급 pclass이 높을수록 생존율이 높다는 것과 남자보다는 여자, 아이보다는 성인(isminor = adult)의 생존율이 높았다는 것을 알 수 있다. 하지만 이들 간의 상호작용은 위 통계만으로는 알기 힘들다. 상호작용이라 하면 승객 등급이 여자와 남자의 생존율 차이에 어떤 영향을 줄 수 있느냐다. 이를 보려면 카이제곱검정chi-square[08]과 같은 통계적 방법이 필요하나, 일반인들에게 이해시키려면 부연설명이 너무 길어지기 때문에 필자의 경우에는 시각화를 이용한다.

그림 1-4 승객등급, 성별, 나이에 따른 생존율

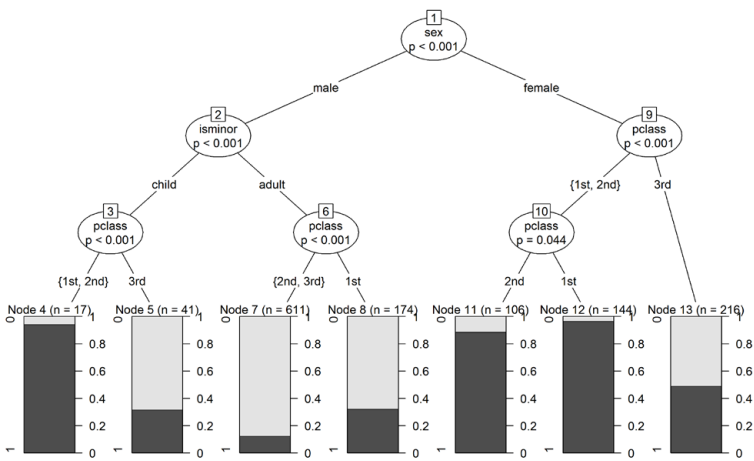


08 카이제곱 독립성 검정을 사용해 특정 변수 사이에 유의미한 관련성이 있는지 검증할 수 있다.

그림 1-4를 보면 Y축에는 성별에 따른 나이를, X축에는 승객등급을 각각 표기했다. 그리고 범례로 오른쪽 끝에 생존율을 확률값으로 넣었다. 그래프 내부에서는 박스 형태로 구간을 나눠 확률 범례를 기준으로 색상의 밝기로 생존율을 표기했다. 이렇게 하면 승객등급과 성별, 성인인지 여부에 따른 생존율을 한눈에 볼 수 있다. 결과 그래프를 보면 승객등급이 올라 갈수록 생존율이 높아지고, 성별과 성인 여부에 영향을 받는 것을 확인할 수 있다.

그림 1-5 그래프는 '결정나무Decision Tree'[09]의 한 알고리즘으로, 데이터를 분석해 생존에 영향을 미치는 인자가 어떤 것인지 나무 모형으로 도식화한 것이다. 이를 보면 일단 성별이 생존에 가장 큰 영향을 끼치며, 이후로는 성별마다 다르고, 남자는 성인인지 아닌지, 여자는 승객등급이 생존에 가장 큰 영향을 끼치는 것을 알 수 있다. 게다가 트리 그래프가 타일그림tile plot 보다는 더 많은 정보를 보여주고 있음을 알 수 있다.

그림 1-5 트리 알고리즘을 이용한 생존율에 영향을 미치는 요인 시각화

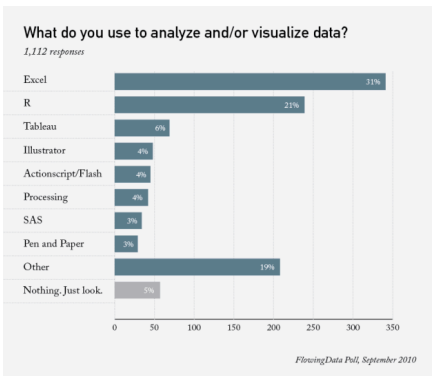


09 결정나무는 데이터마이닝에서 대표적인 분석 방법이다. 결정나무는 간단하게 '의사결정트리' 또는 '결정나무'라 불리기도 한다. 주어진 데이터를 분류하는 목적으로 사용된다.

1.3 앞으로 방향

시각화하는 방법은 굉장히 많다. 'flowingdata.com'에서 시각화에 어떤 툴을 사용하는지 설문을 한 적이 있었는데, 그림 1-6과 같은 결과를 보여줬다.

그림 1-6 flowingdata의 설문 결과



결과를 보면 엑셀과 R이 절반을 넘는다는 것을 알 수 있다. flowingdata 사이트가 시각화에 대한 유명한 사이트라는 것을 감안할 때 유의미한 설문결과[10]다. 사실 엑셀은 훌륭한 도구지만 데이터 분석이라는 측면에서는 한계가 있다. 엑셀은 아주 편한 분석환경을 제공하기 위해 편리한 GUIgraphical user interface와 셀 기반의 데이터 입력과 처리 방식을 추구한다. 이 때문에 사용자들은 어느 셀에서든 함수를 입력하거나 숫자 문자를 입력할 수 있게 되었다. 이이러한 데이터/코드 간의 분리가 힘든 구조가 사용자의 입력 실수를 용인해 분석 에러를 야기할 가능성이 올라간다. 문법이 있는 R과 같은 언어가 가파른 학습 곡선이 있음에도 불구하고 전문 분석가들에게 각광을 받는 이유는 사람이 야기할 수 있는 에러를 언어의 문법체계와 룰이 어느 정도 막아주기 때문이다.

10 "http://flowingdata.com/2010/09/28/poll-results-what-do-you-use-to-analyze-andor-visualize-data/"에서 자세한 설명을 하고 있음

데이터를 시각화하기 위한 단계는 다음과 같다.

① 데이터를 (어떻게든지) 구한다.
② 데이터를 시각화하기 위한 데이터로 변형한다.
③ 데이터를 시각화한다.
④ 시각화 꾸미기(옵션)

위의 과정을 원활히 하기 위해서는 다양한 지식이 필요하다. 그러나 다행히도 ②와 ③은 R이라는 언어에서 탁월한 기능으로 제공하고 있다. 물론 ①은 data.gov, UCI Machine Learning Repository[11] 등과 같은 오픈 데이터 사이트에서 구할 수 있으며, 몇몇 유명한 데이터는 R에서도 제공하고 있으니 큰 걱정은 하지 않아도 된다. ④는 많은 사람이 비싼 상용도구인 어도비 일러스트레이터를 주로 활용하지만 필자는 오픈소스인 '잉크스케이프Inkscape'[12]로 간단하게 편집하곤 한다.

이 책에서는 다음과 같은 내용을 알아볼 것이다.

① R 기본 문법 및 사용법
② R로 데이터 다루기
③ ggplot2를 이용한 R 시각화
④ 벡터 그래프를 이용한 그래프 후처리
⑤ shiny를 이용한 동적 웹 그래픽

이 내용을 따라오기 위해 필요한 사전 지식은 프로그래밍 언어에 대한 경험뿐이다. 프로그래밍 경험이 없더라도 R을 첫 프로그래밍 언어로 배우기엔 손색이 없으니, 의지만 있다면 이 책의 내용을 이해할 수 있을 것이다.

11 http://archive.ics.uci.edu/ml/
12 http://inkscape.org/

2 | R 프로그래밍 기본

이 장에서는 R 언어[01]에 대한 소개와 더불어 프로그래밍 문법을 설명한다. 긴 여정을 떠나면서 꼭 챙겨야 할 가장 중요한 준비물 중 하나이므로 정확히 이해하고 넘어가길 바란다. R은 AT&T에서 개발한 S 언어의 오픈 소스 버전으로 약 20년의 역사를 가진 통계 컴퓨팅statistical computing 언어며, 데이터 시각화를 위한 매우 훌륭한 환경이다.

대표적인 함수형 언어인 '리스프Lisp'에서 파생된 언어로, 함수형 언어의 특징을 포함하고 있으며, 그와 동시에 절차적인 문법구조도 지원한다. 또한 모듈화된 패키지 개발을 위해 객체지향적인 요소들도 포함하고 있다. 특히 함수형 언어이기에 간결한 코드로 구현할 수 있으며, 병렬화 처리 전환이 쉽다.

게다가 인터랙티브 셸shell을 제공해 코드의 결과를 바로 실행·확인할 수 있다. 이러한 특성을 바탕으로 정확하고 빠른 프로그래밍이 가능하다. 인터랙티브 셸을 활용해 기존 함수를 고치거나 새로운 함수를 제작해 루틴화로 반복 작업을 효율적으로 처리할 수 있다.

R은 데이터를 다루는 언어다. 데이터 획득과 조작munging[02], 모델링, 시각화 등은 데이터 분석을 위한 큰 줄기에 해당하는 업무들이다. R은 분석 업무를 구석구석 최고로 다뤄줄 수 있는 툴이자 통합 개발 환경IDE이라고 할 수 있다. 그 환경을 이해하려면 R 언어에 대한 문법 이해가 필요하다. 사실 R은 많은 프로그래밍 언어의 영향을 받아왔다. 특히 리스프[03]라는 함수형 언어의 영향을 많이 받았는데, 이 때문에 많은 R 초보자가 배움에 어려움을 겪고 있다. 하지만 이 덕분에 데이터 분석 시 복잡하

01 참고 문헌 [14]를 참조(R Core Team, 2012)

02 사전에 나오지 않은 신조어로 데이터를 분석 목적에 맞게 조작을 하는 행위를 의미한다. '멍잉'이라고 읽는다.

03 리스프에 대해서 좀 더 알고 싶다면 『만들면서 배우는 리스프 프로그래밍』(한빛미디어, 2011년 11월)을 참고하기 바란다.

고 이해하기 힘들고 유지보수가 어려운 코드를 만들지 않아도 된다.

인터랙티브한 분석은 분석툴의 유연성을 기반으로 한다. 분석 시 사용한 코드는 다른 분석에서 재사용할 수 있어야 하며, 몇 번이고 유사한 분석을 큰 노력 없이 반복적으로 수행할 수 있어야 한다. 분석 데이터의 다양성과 분석 요구의 복잡성 때문에 통계량으로 데이터를 이해하는 것 이상으로 시각화로 데이터를 이해하는 방법이 보편화됐다. 예를 들어 두 개 이상의 정규분포가 결합된 혼합정규분포mixture normal distribution 데이터세트를 기초적인 평균, 분산, 왜도, 첨도 등으로 분석해도 대략적인 얼개만 예측 가능할 뿐이다. '백 번 듣는 것보다 한 번 보는 게 낫다'는 말처럼, 나열된 수치보다 보기 쉽게 시각화한 결과가 훨씬 빠르고 확실한 상황 판단이 가능하다.

이런 인터랙티브한 분석을 수행하는 데 가장 적합한 환경 중 하나가 R이다. R은 수많은 소스에서 데이터를 가져올 수 있고, 최신의 다양한 알고리즘을 적용할 수 있으며, 언어적으로 재활용성을 중시해 CRANComprehensive R Archive Network이라는 패키지 배포 시스템을 갖고 있다. 게다가 분석가들이 많이 사용하는 그래픽 라이브러리를 포함하고 있다.

하둡Hadoop과 같은 분산 컴퓨팅 환경이 없을 때 메인프레임 같은 대형 컴퓨터에서 데이터 분석은 비용이 매우 비쌌다. 때문에 마음 놓고 분석해볼 수 있는 여건이 아니었다. 컴퓨팅 환경과 분석 파라미터를 신중하게 선택·입력해야 했고, 결과 파일도 수백 페이지에 이르렀다. 이런 결과 파일에서 원하는 내용을 선별하고 나머지는 버렸다. 대부분의 통계 함수나 패키지는 이런 대형 컴퓨터 환경일 때 개발됐고, 현재까지 그런 분석 결과를 보여주는 방식을 따르고 있다.

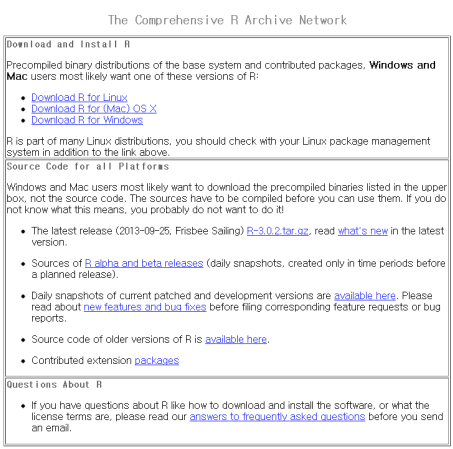
하지만 개인용 컴퓨터가 일반화되면서 예전처럼 한 번에 설정하고 엄청난 양의 분석 결과를 얻는 그런 과정에서 벗어나기 시작했다. 다양한 데이터 입력, 변환, 무응답 대체data imputation, 데이터 추가 및 삭제, 시각화, 모델링 과정 모두를 인터랙티브

하게 수행하면서, 분석가 자신이 갖고 있던 데이터에 대한 질문의 답을 완전히 얻을 수 있는 방법이 일반화됐다. 예를 들어 모델링 퍼포먼스를 확인하기 위해 시각화 과정으로 돌아갈 수 있고 시각화 과정을 위해 데이터 조작 과정이 필요할 수 있는데, 이들 간에는 절차상의 우선순위가 없이 유동적으로 필요에 따라 선택하여 분석할 수 있다.

2.1 R을 설치하자

R은 "http://r-project.org"에서 다운받을 수 있다. R뿐만 아니라 '패키지'라는 라이브러리와 같은 파일을 CRAN에서 관리한다. Perl 언어의 다운로딩 가능 리소스가 CPAN에서 일괄 관리되는 것과 같다. 따라서 R을 설치하려면 첫 페이지 왼쪽 메뉴에서 CRAN 링크를 클릭한다. 이어서 [Korea] 항목으로 이동해 "http://cran.nexr.com"을 선택하면, 운영체제별로 컴파일된 바이너리 또는 소스를 다운받을 수 있다. 계속해서 [Download R for Windows]를 클릭해 나타나는 창에서 [base]를 선택해 R 바이너리를 내려 받는다. 윈도우 바이너리 다운로드 화면은 그림 2-1과 같다.

그림 2-1 R 다운로드 페이지



국내 R 사용자의 90% 이상이 윈도우 운영체제를 사용하고 있다는 통계 자료[04]에 의거해 앞으로 예제는 윈도우 기반으로 소개한다.

CRAN 한국 미러링 서버 중에 "http://cran.nexr.com"은 필자가 구축해서 관리했으며 넥스알에서 후원하고 있다. 하루에 두 번 이상 r-project의 메인 서버와 동기화하기 때문에 대부분의 경우 가장 최신의 파일을 담고 있다. 그럼에도 "http://cran.rstudio.com" 사용을 권한다. RStudio(RStudio에 대해서는 2.3절에서 다룸)의 미러링 서버가 Amazon CloudFront[05]에 구축되어 있어서 한국에서 사용하기에 전혀 불편함이 없고, 무엇보다 서버 로그를 공개적으로 배포[06]하고 있기 때문이다. 이 덕분에 한국 R 사용자들에 대한 통계를 낼 수 있으며, 이런 책을 쓸 때 예상 독자의 수준을 파악할 수 있고 로그 자체가 좋은 데이터 분석 리소스로 사용될 수 있는 장점이 있다.

설치할 때 x64 바이너리만 선택해 설치하기 바란다. 많은 사용자가 64비트 운영체제를 사용하는데, 64비트 OS에서는 x64 바이너리가 가장 최적으로 작동하기 때문이다. 특히 운영체제에서 메모리를 4GB 이상 사용한다면 반드시 x64 바이너리를 사용해야 한다. 물론 x86 바이너리를 함께 설치해도 작동에는 문제될 게 없지만, 사용하지 않을 바이너리까지 설치하면 시스템이 복잡해지고 공간도 낭비된다. 설치과정에서는 [32-bit Files]를 체크 해제하면 x64 바이너리만 설치된다. 물론 32비트 OS를 사용한다면 64비트 바이너리 선택 화면은 나오지도 않는다.

소스코드를 통해 직접 빌드해 보는 것도 R이 어떻게 동작하는지 이해할 수 있는 좋은 방법이지만, 그 과정이 간단하지 않고 본 장의 주제와 직결되지 않아 여기서는 생략한다. 하지만 호기심 많은 독자라면 직접 한 번 시도해 보기 바란다.[07]

---

04 http://freesearch.pe.kr/archives/2672

05 http://aws.amazon.com/ko/cloudfront

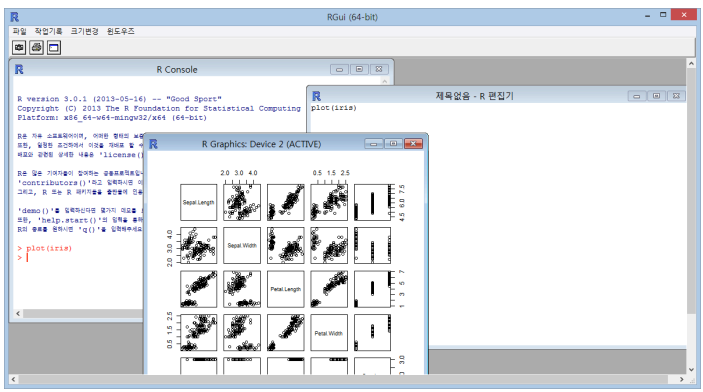
06 http://cran-logs.rstudio.com/

07 R은 데비안 계열의 리눅스에서 개발되고 있다. 따라서 리눅스 계열에서 가장 잘 동작한다. 특히 우분투

2.2 R GUI와 IDE 소개

그림 2-2는 일반적인 윈도우 R 콘솔 화면이다. R을 설치한 후에 실행한 화면을 보면 썰렁하기 그지 없다. 단순히 코드 에디터와 인터랙티브 셸, 그리고 가끔 팝업되는 help 페이지나 그래프 창이 전부이기 때문이다. 이 때문에 R을 처음 사용하는 사용자들이 낯설어한다. 따라서 좀 더 편리한 시스템이 없는지 알아보자.

그림 2-2 윈도우용 R 실행 화면



R을 손쉽게 사용하게 하기 위한 공개된 IDE[Integrated Development Environment]와 GUI[Graphical User Interface] 환경을 알아보자. GUI 환경은 다음과 같다.

- Poor Man's GUI(pmg)
- Jaguar: Java GUI for R
- R commander
- Rattle GUI

계열에서는 특별한 설정 없이 'sudo apt-get install r-base' 명령어로 손쉽게 R을 설치할 수 있다. 레드햇 계열도 yum으로 설치할 수 있지만, 별도의 저장소(repository)를 설정해야 한다. 리눅스에 익숙한 사용자라면 리눅스에서 R 사용을 추천한다. 이는 R 코딩 용이성 측면뿐 아니라 리눅스 서버를 사용할 시 꽤 쾌적하게 서버 리소스를 사용할 수 있는 자유로움도 따르기 때문이다.

만일 R의 연산 퍼포먼스를 튜닝하고 싶다면 반드시 리눅스 환경이 필요하다. yum이나 apt-get을 이용한 설치 버전보다 직접 빌드한 R의 퍼포먼스가 더 좋으며, 수치 연산 라이브러리(BLAS)에 따른 행렬 연산 퍼포먼스의 경우 많게는 수십 배 이상 차이가 난다. 게다가 gcc가 아닌 인텔 컴파일러를 사용할 수 있다면 퍼포먼스는 더 좋아진다.

이 중에서 가장 널리 사용되는 GUI는 R commander[08]로, 설치 방법은 다음과 같이 R Console(콘솔)에서 입력한다.

```
#설치
install.packages("Rcmdr")

#실행
library(Rcmdr)
```

실행 중에 여러 패키지가 함께 설치됨에 따라 다소 시간이 걸릴 수 있다. R 실행 환경에 익숙하지 않다면 메뉴 방식으로 명령어를 생성해주는 이와 같은 GUI를 기반으로 접근해보는 것도 나쁘지 않다. 하지만 이런 툴을 사용하면서 명심해야 할 부분은 실제 명령어가 어떻게 생성되는지 확인하고 기억해야 한다는 것이다. 그런 과정을 통해서 R이 동작하는 방식을 이해하게 되며, 좀 더 유연하게 R을 사용할 수 있기 때문이다.

08 참고 문헌 [5]를 참고(Fox J, 2005)

Rattle GUI는 데이터마이닝에 특화된 환경을 제공하며, 업무가 데이터마이닝이라면 이 환경을 한 번 사용해 볼 것을 권한다. 이 툴을 기반으로 데이터마이닝을 설명한 도서[09]가 있다. 툴 설명과 더불어 R을 기반으로 데이터 마이닝하는 좋은 예제들이 있는 책이니 관심 있는 독자는 읽어보길 바란다.

```
#설치
install.packages("rattle")

#패키지 로딩
library(rattle)

#실행
rattle()
```

GUI 환경에서 R 명령어들이 익숙해진 후라면 적절한 IDE를 구하게 되는데, 현재 많이 쓰이는 IDE는 다음과 같다.

- RStudio
- RKward
- Eclipse with StatET
- Tinn R

상용 R IDE로 Revolution R[10]이 있는데 학생들에게 아카데미 버전을 무료로 제공하고 있으니 관심 있는 독자는 사용해 봐도 좋을 것이다.

09 참고 문헌 [19]를 참조(Williams GJ, 2011)

10 "http://www.revolutionanalytics.com/products/revolution-r.php"에서 배포한다. MS의 컴파일러로 컴파일한 R 환경을 제공하며, 행렬과 같은 수치 연산에 최적화된 상용 라이브러리를 포함하고 있어 복잡한 수치연산을 요하는 작업을 한다면 적절한 대안이 될 수 있다.

2.3 RStudio를 이용하자

처음부터 R 기본 셸을 사용하는 것도 나쁘지 않지만, RStudio[11]를 사용하면 최적의 R 사용 환경을 유지할 수 있기 때문에 정말 편하다. RStudio는 그 자체만으로도 많은 기능을 포함하고 있어서 이곳에서 전체를 소개하기는 쉽지 않다. 일단 필자가 알고 있는 RStudio의 장점은 다음과 같다.

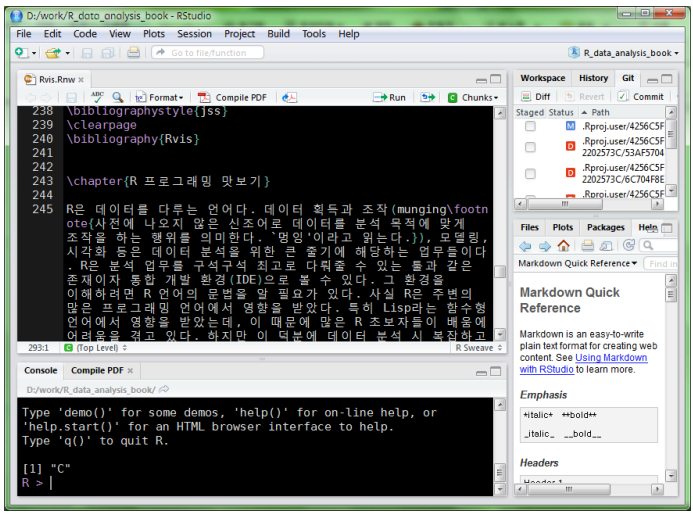
- 에디터, 콘솔, 명령어 히스토리, 시각화, 파일탐색, 패키지 관리 등을 한 화면에서 보여준다.
- 프로젝트의 관점으로 파일 관리를 해준다. 물론 소스코드 관리 시스템과 연계할 수 있다.
- 빌트인 데이터 뷰어 내장, 플로팅 히스토리, R help 결합, Sweave[12]와 knitr[13]가 통합돼 있다.
- R Markdown[14] 내장으로 문서와 코드를 결합할 수 있게 하고, 재현성 있는 분석을 가능하게 한다.
- 패키지 빌드 자동화와 Rcpp[15] 편집 환경을 제공한다
- 리눅스, 맥, 윈도우 등 멀티 플랫폼을 지원한다.

무엇보다 RStudio는 R 커뮤니티 내에서 가장 인기 있는 IDE다. 게다가 클라이언트/서버 환경으로 설정해두면 풍부한 서버 리소스를 원격에서 활용할 수 있다. 특히 웹 브라우저 자체가 편집기가 되어 웹 브라우저를 닫아도 최신의 작업 히스토리와 환경을 간직하고 있다. 웹 브라우저와 인터넷만 있으면 어느 곳에서든지 자신이 작업한 환경을 그대로 사용할 수 있다. 필자는 수백 GB의 메모리를 가진 서버에 RStudio를 설치해두고 리소스가 풍부하지 않은 노트북에서 원격으로 접속해 사용

11 http://www.rstudio.org/
12 참고 문헌 [12]를 참조(Leisch, 2002)
13 참고 문헌 [20]을 참조(Xie, 2013)
14 참고 문헌 [1]을 참조(Allaire et al., 2013)
15 참고 문헌 [4]를 참조(Eddelbuettel and François, 2011)

하는데 이는 정말 최고의 작업 환경이다.[16]

그림 2-3 윈도우용 RStudio 화면 예



16 R은 모든 데이터를 메모리에 올려놓아야만 분석이 가능하다. 일단 메모리에 데이터를 올리려면 시스템 메모리가 충분해야 된다. 게다가 대부분의 분석 업무는 같은 데이터를 쪼개고 붙이고 하는 작업의 연속이므로 필요 메모리량은 더 늘어난다. 따라서 필자는 업무를 하면서 메모리에 대한 고민을 하고 싶지 않아 될 수 있는 한 최고 스펙의 서버에서 분석을 수행한다. 단순히 메모리만 많다고 좋은 것도 아니다. 만일 큰 데이터를 멀티코어 프로세싱으로 처리하고 싶다고 한다면 CPU 스펙도 메모리 못지 않게 좋아야 된다.

2.4 R 프로그래밍

2.4.1 패키지 설치 및 도움 얻기

이미 앞에서 명령어를 사용해 봤겠지만, 패키지를 관리하는 방법은 꼭 숙지해야 한다. 명령어 설명 이전에 CRAN 미러링 서버가 적절하게 설정돼 있는지 확인이 필요하다. 미러링 서버는 중앙 CRAN 서버의 모든 데이터를 복사해 지리적으로 가까운 사용자에게 쾌적하게 패키지 관리를 할 수 있게 도와주는 서버다.[17]

RStudio에서 [Tools]-[Options]-[Packages] 메뉴에 들어가서 CRAN mirror[18]가 NexR이나 중앙대학교 혹은 Global(CDN)-RStudio로 설정돼 있는지 확인하고 설정하면, 미러링 서버로 해당 서버가 사용된다. 만일 이 부분이 적절하게 설정되지 않았다면 패키지 다운로딩과 설치에 다소 시간이 걸릴 수 있다.

예를 들어 필자가 만든 KoNLP[19]라는 한글 텍스트 분석 패키지를 설치하고 싶다면, 다음과 같은 명령어를 실행하면 미러링 서버에서 해당 패키지를 다운받아 설치까지 자동으로 수행된다.

```
install.packages("KoNLP")
```

현재 패키지가 어떤 것들이 설치되어 있는지 확인하고 싶으면 RStudio 오른쪽 메뉴창에서 [Packages] 항목을 선택해 리스트를 확인해 보거나 콘솔창에서 다음과 같은 명령어를 입력하면 설치된 리스트가 출력된다.

```
installed.packages()
```

---

17 2013년 현재 한국에서 NexR과 중앙대학교에서 서버를 제공하고 있다.

18 "http://cran.r-project.org/mirmon_report.html"에서 미러 서버가 얼마나 업데이트를 잘하고 있는지 확인 가능하다. 만일 패키지 설치나 관리가 생각한대로 동작하지 않는다면 자신의 미러 서버가 현재 잘 동작하고 있는지 이 페이지에서 확인해 볼 수 있다.

19 참고 문헌 [10]을 참조(Jeon, 2012)

R의 전체 패키지 수는 글을 쓰는 현재 4,700여 개[20]가 넘으며 각종 패키지의 업데이트가 자주 이뤄지는 편이다. 따라서 주기적으로 패키지를 업데이트해야 하는데, RStudio에서는 [Tools]-[Check for Package Updates]를 클릭해 업데이트 여부를 체크하여 업데이트하면 된다. 그리고 다음 명령어로 패키지를 체크하면 업데이트 가능한 패키지들을 리스트업 해준다.

```
old.packages()
```

체크와 더불어 업데이트까지 설치하는 명령어는 다음과 같다.

```
update.packages()
```

때로는 apriori 알고리즘을 공부하면서 R에서 구현체를 찾아볼 생각을 할 수도 있을 것이다. 그럴 때 관련 함수나 구현체가 어디에 있는지 알아보기 위해서 고민할 수도 있는데, R에서는 이런 함수나 구현체를 찾기 위한 여러 명령어를 제공한다.

```
#help 시스템에서 해당 단어를 찾아준다.
help.search("apriori")

# 혹은
??apriori
```

위 명령어는 같은 역할을 하지만, 해당 단어가 현재 R help 시스템에 텍스트로 존재해야 한다는 단점이 있다. 쉽게 이야기하면, 사용자의 로컬 R 시스템에 해당 구현체가 설치돼 있어야 한다. 만일 함수나 구현체가 사용자의 로컬 R 시스템에 없다면 아무 검색 결과가 나오지 않는데, 이럴 경우에 사용할 수 있는 명령어가 RSiteSearch()다. 이 명령어는 "http://search.r-project.org"의 검색 결과를 리턴해준다.

```
RSiteSearch("apriori")
```

20 "http://cran.r-project.org/web/packages"에서 패키지 숫자를 확인해 볼 수 있다.

RSiteSearch() 함수의 결과를 테이블 형태로 입력 받아 좀 더 정리된 형태로 웹 브라우저를 통해 보여주는 sos 패키지[21]도 있다. 하지만 자세한 결과를 보고 싶다면 RSiteSearch() 함수 사용을 추천한다.

```
install.packages("sos")
library(sos)
findFn("apriori")

# 혹은

??apriori
```

패키지를 설치하고 로딩했으나 어떤 함수가 있는지 궁금할 때는 다음 명령어를 사용한다.

```
help(package = "arules")
```

2.4.2 R 기본 연산

R을 실행하면 R 콘솔이 뜨면서 메시지와 더불어 '〉'와 같은 콘솔 프롬프트가 나타난다. 대부분의 인터프리터 언어는 이런 형태를 띄고 있으며, 이 문자 뒤에 명령어를 입력하고 엔터키를 누르면 해당 라인의 명령어를 수행하고 바로 평가 결과를 리턴한다.

```
1 + 2 + 3
## [1] 6
1/2
## [1] 0.5
(1 + 2) * 3
## [1] 9
```

21 참고 문헌 [6]을 참조(Graves et al., 2012)

각 명령 결과의 머리를 보면 '[1]'이라는 문자가 함께 출력되는데, 이 문자는 R에서 사용하는 벡터vector라는 자료형 길이의 인덱스를 의미한다. 벡터는 R에서 기본 자료형으로서 '3'이라는 숫자 하나도 길이가 1인 벡터형으로 인식해 처리한다. 위의 결과들은 모두 길이가 하나인 결과를 리턴함으로 '[1]'이라는 인덱스 번호만 뜬다. 이 인덱스 번호는 출력한 벡터 결과의 크기를 가늠하거나 특정 인덱스 번호를 알아야 될 경우 매우 유용하다.

```
1:100
 ##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
 ## [29] 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
 ## [57] 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
 ## [85] 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```

':' 는 첫 번째 숫자에서 시작하고 마지막 숫자로 끝나는 숫자형 벡터를 리턴해 준다. 위 결과에서는 연속된 값을 출력해서 유추하기 쉽기에 필요 없을지 모르지만, 다소 복잡한 결과물일 경우 이런 인덱스 번호는 굉장히 편리하다. 결과 각 라인의 첫 번째 인자의 인덱스 번호를 다음과 같이 출력해 준다.

```
c(1,2,3,4,5)
 ## [1] 1 2 3 4 5

 # 위와 같이 긴 벡터를 생성하기 위해서는 ' c ' concatenate 함수를 사용하면 된다.
 c(1,2,3,4,5) + c(5,4,3,2,1)
 ## [1] 6 6 6 6 6

 c(1,2,3,4,5) * c(5,4,3,2,1)
 ## [1] 5 8 9 8 5
```

```
c(1,2,3,4,5) / c(5,4,3,2,1)
## [1] 0.2  0.5  1.0  2.0  5.0

c(1,2,3,4,5) - c(5,4,3,2,1)
## [1] -4 -2 0 2 4
```

길이가 같은 벡터 간의 연산은 우리가 알고 있는 수학의 벡터 연산과 같은 방식으로, 각 벡터의 인덱스 쌍 사이의 연산을 리턴하면 이를 다시 벡터로 만들어 같은 길이의 벡터가 리턴된다. 만일 길이가 다르다면 어떻게 될까?

```
1 + c(1,2,3,4,5)
## [1] 2 3 4 5 6

c(1,2) + c(1,2,3,4,5)
## [1] 2 4 4 6 6
## 경고 메시지
## In c(1, 2) + c(1, 2, 3, 4, 5)
## Warning: 두 객체의 길이가 서로 배수관계에 있지 않습니다.

c(1,2) + c(1,2,3,4,5,6)
## [1] 2 4 4 6 6 8
```

위의 결과를 보고 예측해 보자. R은 길이가 다른 벡터 간의 연산은 길이가 짧은 벡터를 반복 사용해 연산하고 결과를 리턴한다. 위의 두 번째는 에러가 나는데, 이는 길이가 긴 벡터가 작은 벡터 길이의 배수가 아니기 때문이다. 그러나 마지막 결과는 정확하게 도출됨을 볼 수 있다. 이런 연산방법은 R을 사용하면서 상당히 많은 부분에서 편리하게 활용되니 기억해 두기 바란다.

숫자뿐만 아니라 문자열도 벡터에 저장된다.

```
# 이후의 '#' 문자열은 주석입니다.

"안녕하세요. 고감자입니다."
## [1] "안녕하세요. 고감자입니다."

c("안녕하세요", "고감자입니다.")
## [1] "안녕하세요"  "고감자입니다."
```

2.4.3 함수

R에서 실제 작업을 수행하는 건 함수Function다. 이미 이전 코드에서 수많은 함수를 사용하였으며, 프로그래밍에 익숙한 독자라면 함수가 어떤 형태로 구성되는지 이미 익숙해졌을 것이다.

```
foo(arg1, arg2, ...)
```

함수의 형태는 위와 같은 형태며 'arg1'은 함수에 넘겨지는 첫 번째 인자, 'arg2'는 두 번째 인자다.

```
sqrt(2)
## [1] 1.414

abs(-24)
## [1] 24
```

sqrt는 제곱근을 구하는 함수며, 반환값으로 입력 인자로 넘겨진 숫자의 제곱근을 반환한다. abs는 입력값의 절대값을 반환하는 함수다. 좀 더 자세히 들어가면 '+, -' 등의 연산자도 함수의 일환이다.

```
1+2
## [1] 3

 ‘ + ’ (1,2)
## [1] 3
```

위 두 예를 제외하고도 생각보다 많은 부분이 함수로 이루어져 있다. 다만 우리가 사용하는 표현은 대부분이 ‘달콤한 문법syntactic sugar’[22]이다. 사실 “2.4.2 R 기본 연산”에서 설명한 연산자 모두 함수인데, 그만큼 R의 많은 부분이 함수로 이뤄져 있다.

그럼 함수를 만드는 방법을 알아보자.

```
foo <- function(arg1, arg2, ..., bar = bar_val){
    print(arg1)
    print(arg2)
    print(1231232323131231, ...)
    print(bar)
    return(1)
 }
ret <- foo("첫 번째", "두 번째", 5, bar = "바")
## [1] "첫 번째"
## [1] "두 번째"
## [1] 1.2312e+15
## [1] "바"

ret
## [1] 1
# 함수의 리턴값이 1이다.
```

22 컴퓨터가 쉽게 알아들을 수 있는 문법과 동일한 기능을 수행하는 사람들이 이해하기 쉬운 문법들을 의미한다. `+` 함수를 두 개의 인자를 받는 함수로 컴퓨터는 인지하지만 사용자는 그저 `+`를 함수라기 보다는 연산자로 인식해 사용하고 이 역시 잘 동작하는데 이를 달콤한 문법이라고 칭한다.

함수는 대부분 위와 같은 형태를 띤다. 'arg1, arg2'의 경우 첫 번째 인자 두 번째 인자를 의미하며 '...' 는 대부분 함수 내부에서 다른 함수를 호출할 때 해당 함수에 추가 인자를 넘겨주기 위해서 사용한다. '...' 이후에는 반드시 명명된 변수만 올 수 있다. 위에서는 'bar ='와 같이 인자를 넘겨줄 때 이름을 명명해줘야 된다.

2.4.4 변수

객체에 값을 저장하기 위해서 R 콘솔에서는 '<-' 혹은 '->' 연산자를 사용한다. 물론 다른 언어에서는 '=' 연산자를 주로 사용하나, 이 책에서는 '같다(==)'라는 의미와 혼동을 할 수 있어서 '<-'를 사용하겠다. 단, 함수에서 인자에 값을 넣을 때는 '<-'는 동작하지 않으며 '='를 사용해야 된다.

```
x <- 1
x
## [1] 1
```

첫 라인은 "x 변수는 1을 얻었다(gets)"라고 이해하면 되며, 두 번째 라인은 "x에 어떤 값이 들어 있는지 평가한다"라고 이해하면 된다. x는 그 자체로 심볼Symbol이며 1이라는 객체Object를 지칭하고 있다.

```
x <- 3
y <- 5
z <- c(x, y)
z
## [1] 3 5
y <- 200
z
## [1] 3 5
```

```
y
## [1] 200
```

'z'에 저장되는 순간 값이 복사되며, 이후에 'y' 값이 변경되더라도 'z' 값은 그대로 유지된다. c 함수가 평가될 당시 x, y 값이 복사돼 z 값으로 리턴되기 때문이다. R 함수는 기본적으로 값에 의한 호출call by value을 적용하고 있으며, 별도로 트릭을 이용해 '참조에 의한 호출call by reference'을 구현할 수 있다. 참조에 의한 호출은 공식적으로 지원하지 않는 기능이므로 이 기능을 사용하고 싶다면 R 환경에 대한 이해가 있어야 한다. 내부적으로는 참조에 의한 호출을 활용하기도 하는데, 인자로 매우 큰 객체를 넘길 때 무조건 객체 복사를 수행하게 되면 굉장히 오랜 시간이 소요된다. 만일 이런 객체를 읽기 용도로만 활용한다면, 함수 내부적으로 객체 복사를 하지 않고 참조에 의한 호출로 객체를 사용하게 된다.

2.4.5 벡터

이미 벡터Vector는 지금까지 심심치 않게 사용해 왔다. 벡터는 단 한 종류의 자료형만을 저장할 수 있는 자료구조다. 이 자료형들에는 정수형integer, 부동소수점수floating-point number, 복소수complex number, 문자열text, 논리값logical value, 로raw 등이 올 수 있다. 벡터에는 한 가지 자료형만 존재할 수 있기 때문에 다음과 같이 자료형 강제변환coerces이 수행되기도 한다. 'typeof()' 함수는 변수의 저장타입을 리턴한다.

```
v <- c(1, 2, 3, 4, 5, 6)
v
## [1] 1, 2, 3, 4, 5, 6
typeof(v)
## [1] " double "

coerced <- c(1, 2, 3, 4, 5, 6, "고감자")
coerced
```

```
## [1] "1"  "2"  "3"  "4"  "5"  "6"  "고감자"

typeof(v)
## [1] " double "

typeof(coerced)
## [1] " character "
```

이런 변환을 거치면 데이터의 손실이 없는 가장 일반화된 자료형으로 변환된다. 이어서 벡터형 변수를 정의하고 특정 값들을 가져올 수 있는 방법들을 소개한다.

```
vec <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
# 첫 번째 원소
vec[1]
## [1] 1

#첫 번째 원소를 다른 숫자로 대체
vec[1] <- 100

#홀수 인덱스에 있는 값들만
vec[c(TRUE, FALSE)]
## [1] 100   3   5   7   9

#짝수 인덱스에 있는 값들만
vec[c(FALSE, TRUE)]
## [1] 2   4   6   8   10
```

```
#첫 번째부터 세 번째 값
vec[1:3]
## [1] 100   2   3

#3의 배수
vec[vec %% 3 == 0]
## [1] 3   6   9

#3의 배수가 아닌 것들
vec[!(vec %% 3 == 0)]
## [1] 100   2   4   5   7   8   10

#세 번째, 열 번째 인자를 제거한 벡터
vec[-c(3,10)]
## [1] 100   2   4   5   6   7   8   9
```

위와 같은 특정 데이터 추출 방법은 R 기반 데이터 멍잉munging 전반에서 사용되기 때문에 반드시 익혀둬야 된다. 두 벡터를 결합하는 방법은 다음과 같다.

```
v <- 1:10
z <- 10:1
comb <- c(v, z)
comb2 <- append(v, z)

comb
## [1] 1 2 3 4 5 6 7 8 9 10 10 9 8 7 6 5 4 3 2 1

comb == comb2
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [15] TRUE TRUE TRUE TRUE TRUE TRUE
```

':'는 시작 숫자에서 끝 숫자까지 1씩 증가하거나 감소하는 숫자형 벡터를 반환해 주는데, 때로는 증가방식을 달리 하고 싶을 때가 있을 것이다.

```
seq(from = 1,to = 10, by = 2)
## [1] 1 3 5 7 9
# 벡터의 길이는 ' length() ' 함수를 통해 알 수 있다.

v <- c(1:20)
length(v)
## [1] 20

length(v) <- 30    # 벡터 v의 길이가 30인 정보를 입력한다.
v
## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 NA NA NA NA NA NA NA NA
## [29] NA NA

length(v) <- 10
v
## [1] 1 2 3 4 5 6 7 8 9 10
```

원래 길이가 20인 벡터에 길이가 30인 정보를 넣을 경우 나머지 슬롯에는 NA(Not Available) 값이 들어간다. 'NA'는 값이 없음을 의미하며, 데이터 분석업무에서는 주로 손실된 값을 지칭한다. 그러나 실제 벡터 길이보다 작은 값을 길이 정보로 넣을 경우, 길이가 넘는 범위에 있는 데이터는 제거된다.

때때로 길이가 결정된 벡터를 미리 만들어야 한다. R과 같은 인터프리터 언어들은 대부분 두 벡터를 결합하거나 원소 하나를 추가하는 작업을 수행할 때 매번 새로운 벡터를 만들게 된다. 내부적으로는 메모리 영역을 할당하고 초기화하는 등 많은 부가적인 연산을 수행하는데, 이런 작업이 많을 때는 사용할 충분한 크기의 벡터를